Dark Mode Implementation

Play+ Dark Theme: Automated, Accessible, and On-Brand Introduction ■ In the Play+ ecosystem, a smooth, comfortable experience in all environments is a core design goal. Whether your users prefer a darker UI or are working late into the evening, Play+ adapts effortlessly. Our Dark Theme isn't bolted on—it's built in, and designed to maintain your brand's distinctiveness and readability. You don't need to define a separate dark theme. Just define your light theme as usual, and Play+ takes care of the rest. Why There's No Separate Dark Theme ■ Traditionally, dark mode meant duplicating styles, increasing complexity and potential bugs. In Play+, that duplication is unnecessary. With a single _dark.css file, Play+ intelligently derives a dark variant of your theme. It respects your brand's tone and automatically adjusts colors, backgrounds, and contrast to suit dark contexts—without disrupting your design language. How It Works ■ Define Your Theme Provide your core tokens in _default.css , including brand colors, text styles, and backgrounds. Dark Theme Engine Kicks In Play+ processes this theme and generates a full dark mode version via _dark.css . No extra configuration required. System Preference Detection When a user's system is set to dark mode, Play+ switches automatically using the prefers-color-scheme: dark media query. What Gets Transformed? ■ Surfaces ■ Light backgrounds are softened to rich dark grays (e.g., --global-color-gray-900), avoiding pure black. Secondary layers maintain visual depth. /* Light Theme */ --color-background-primary : var (--global-color-white) ; /* Dark Theme */ --color-background-primary : var (--global-color-gray-900) ; Text ■ Text colors are lightened to remain readable on dark surfaces, and accessibility contrast is recalculated. /* Light Theme */ --color-text-primary : var (--global-color-gray-700) ; /* Dark Theme */ --color-text-primary : var (--global-color-gray-100) ; Brand Colors ■ Bright brand colors are adapted—desaturated or brightened if needed—to reduce harsh contrast. Related tokens like --color-text-on-brand-primary adjust accordingly. /* Light Theme */ --color-brand-primary : var (--global-color-pink-500) ; --color-brand-secondary : var (--global-color-blue-500) ; /* Dark Theme */ --color-brand-primary : var (--global-color-pink-300) ; --color-brand-secondary : var (--global-color-blue-300) ; Disabling Automatic Detection ■ To ignore system preferences and apply themes manually, update your config: /* Disable automatic theme switching */ [data-theme = "light"] { /* Force light theme */ } [data-theme = "dark"] { /* Force dark theme */ } This disables automatic switching. You can then manage the theme explicitly via toggle or app logic. Overriding the Defaults ■ Most themes work great with automatic derivation. But if you need to override a specific value, just add a custom token in your theme file: [data-theme = "dark"] { /* Override specific dark theme values */ --color-brand-primary : #5aacff ; --color-background-primary : #0a0a0a ; --glass-background-color : rgba (0 , 0 , 0 , 0.8) ; } This gives you precise control when needed—without losing the benefits of derivation. Manual Theme Toggle ■ You can give users a manual theme toggle in your UI using the data-theme attribute on the <html> element. This is especially useful if you've disabled automatic OS detection. For Angular ■ Add this logic to a shared service or component, such as theme-toggle.component.ts : // src/app/theme-toggle/theme-toggle.component.ts export class ThemeToggleComponent { toggleTheme ( ) { const root = document . documentElement ; const isDark = root . getAttribute ("data-theme") === "dark" ; if (isDark) { root . removeAttribute ("data-theme") ; } else { root . setAttribute ("data-theme" , "dark") ; } } } Template: <!-- theme-toggle.component.html --> < button (click) = " toggleTheme() " > Toggle Theme </ button > File Summary ■ Angular : Implement in a dedicated theme-toggle.component.ts with corresponding HTML Tip: You can persist user preference using localStorage if desired. You can give users a theme toggle in your UI using the data-theme attribute on <html> : function toggleTheme ( ) { const root = document . documentElement ; const isDark = root . getAttribute ("data-theme") === "dark" ; isDark ? root . removeAttribute ("data-theme") : root . setAttribute ("data-theme" , "dark") ; } This empowers users and complements OS-level preference detection. Developer Checklist ■ Define a complete light theme in _default.css Let Play+ derive the dark variant automatically via _dark.css Preview before overriding Use custom overrides sparingly Confirm contrast accessibility if overridden Offer a user toggle if needed Conclusion ■ With Play+, dark mode is automatic, accessible, and brand-aware. There's no need to manage two sets of styles or worry about visual quality. One well-defined theme is all it takes to deliver a polished experience—day or night.